In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
house=pd.read_csv("C:/Users/TERIAK-JB/Desktop/GOMYCODE/kc_house_data.csv",encoding="iso-8859-1")
```

In [3]:

```
house.head(10)
```

Out[3]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_abov
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	118
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	217
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	77
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	105
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	168
5	7237550310	20140512T000000	1225000.0	4	4.50	5420	101930	1.0	0	0	...	11	389
6	1321400060	20140627T000000	257500.0	3	2.25	1715	6819	2.0	0	0	...	7	171
7	2008000270	20150115T000000	291850.0	3	1.50	1060	9711	1.0	0	0	...	7	106
8	2414600126	20150415T000000	229500.0	3	1.00	1780	7470	1.0	0	0	...	7	105
9	3793500160	20150312T000000	323000.0	3	2.50	1890	6560	2.0	0	0	...	7	189

10 rows × 21 columns

In [4]:

```
house.shape
```

Out[4]:

```
(21613, 21)
```

In [5]:

```
house.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             21613 non-null  int64  
 1   date           21613 non-null  object 
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64  
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64  
 6   sqft_lot       21613 non-null  int64
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64
 9   view           21613 non-null  int64
 10  condition      21613 non-null  int64
 11  grade          21613 non-null  int64
 12  sqft_above     21613 non-null  int64
 13  sqft_basement  21613 non-null  int64
 14  yr_built       21613 non-null  int64
 15  yr_renovated   21613 non-null  int64
 16  zipcode        21613 non-null  int64
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64
 20  sqft_lot15     21613 non-null  int64
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

In [6]:

```
house.drop("yr_renovated",1,inplace=True)
```

In [7]:

```
house["bathrooms"].apply(np.round)
house["bathrooms"]=house["bathrooms"].apply(np.int)
```

In [8]:

```
house=house.drop(house[(house.bedrooms>10)].index)
```

In [9]:

```
house
```

Out[9]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade
0	7129300520	20141013T000000	221900.0	3	1	1180	5650	1.0	0	0	3	7
1	6414100192	20141209T000000	538000.0	3	2	2570	7242	2.0	0	0	3	7
2	5631500400	20150225T000000	180000.0	2	1	770	10000	1.0	0	0	3	6
3	2487200875	20141209T000000	604000.0	4	3	1960	5000	1.0	0	0	5	7
4	1954400510	20150218T000000	510000.0	3	2	1680	8080	1.0	0	0	3	8
...	...	...	...	...	...	...	...	...	...	...	...	...
21608	263000018	20140521T000000	360000.0	3	2	1530	1131	3.0	0	0	3	8
21609	6600060120	20150223T000000	400000.0	4	2	2310	5813	2.0	0	0	3	8
21610	1523300141	20140623T000000	402101.0	2	0	1020	1350	2.0	0	0	3	7
21611	291310100	20150116T000000	400000.0	3	2	1600	2388	2.0	0	0	3	8
21612	1523300157	20141015T000000	325000.0	2	0	1020	1076	2.0	0	0	3	7

21611 rows × 20 columns

In [10]:

```
house["floors"].apply(np.round)
house["floors"]=house["floors"].apply(np.int)
```

In [64]:

```
house
```

Out[64]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	sqft_above	sqft_
0	7129300520	20141013T000000	221900.0	3	1	1180	5650	1	3	7	1180	
1	6414100192	20141209T000000	538000.0	3	2	2570	7242	2	3	7	2170	
2	5631500400	20150225T000000	180000.0	2	1	770	10000	1	3	6	770	
3	2487200875	20141209T000000	604000.0	4	3	1960	5000	1	5	7	1050	
4	1954400510	20150218T000000	510000.0	3	2	1680	8080	1	3	8	1680	
...	...	...	...	...	...	...	...	...	...	...	...	
21608	263000018	20140521T000000	360000.0	3	2	1530	1131	3	3	8	1530	
21609	6600060120	20150223T000000	400000.0	4	2	2310	5813	2	3	8	2310	
21610	1523300141	20140623T000000	402101.0	2	0	1020	1350	2	3	7	1020	
21611	291310100	20150116T000000	400000.0	3	2	1600	2388	2	3	8	1600	
21612	1523300157	20141015T000000	325000.0	2	0	1020	1076	2	3	7	1020	

21268 rows × 18 columns

In [12]:

```
house["waterfront"].value_counts()
```

Out[12]:

```
0    21448
1      163
Name: waterfront, dtype: int64
```

In [13]:

```
house.drop("waterfront",1,inplace=True)
```

In [14]:

```
house["view"].value_counts()
```

Out[14]:

```
0    19487
2      963
3      510
1      332
4      319
Name: view, dtype: int64
```

In [15]:

```
house.drop("view",1,inplace=True)
```

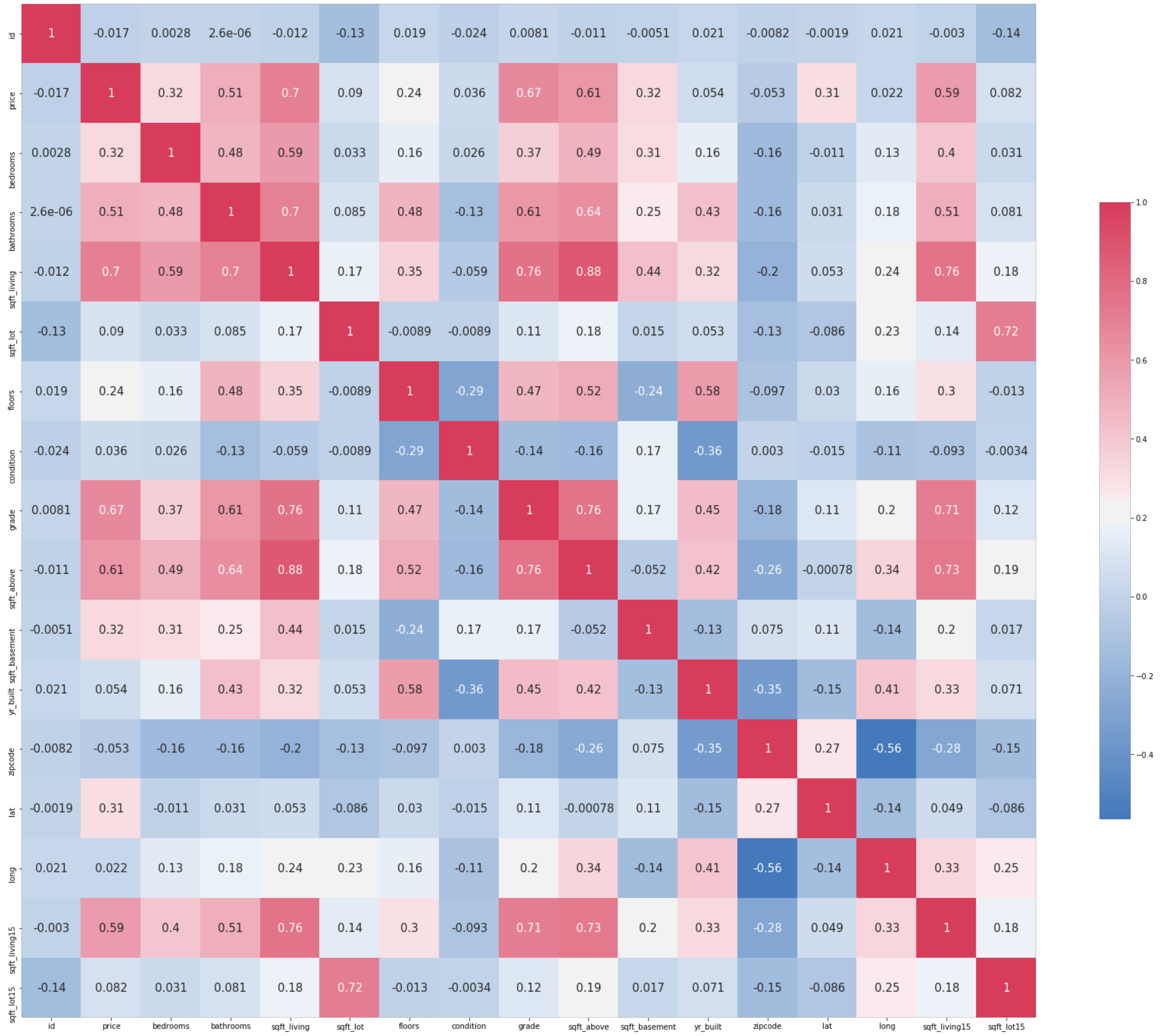
In [16]:

```
def plot_correlation_map( df ):
```

```
def plot_correlation_map( df ):
    corr = df.corr()
    s , ax = plt.subplots( figsize =( 30 , 30 ) )
    cmap = sns.diverging_palette( 250 , 5 , as_cmap = True )
    s = sns.heatmap(
        corr,
        cmap = cmap,
        square=True,
        cbar_kws={ 'shrink' : .50 },
        ax=ax,
        annot = True,
        annot_kws = { 'fontsize' : 15 }
        )
```

In [17]:

```
plot_correlation_map( house )
```



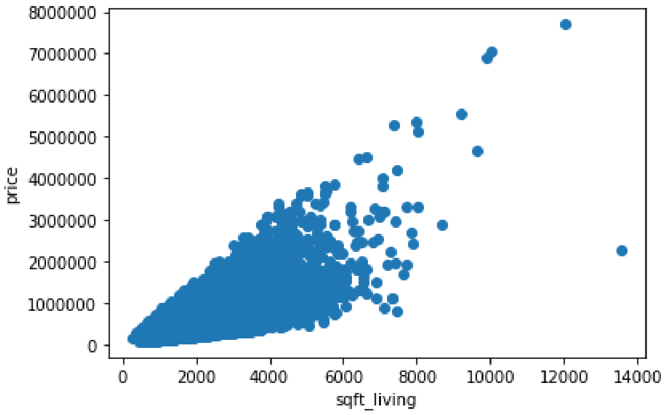
	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	sqft_above	sqft_basement	yr_built	zipcode	lat	long	sqft_living15	sqft_lot15
id	1	-0.017	0.0028	2.6e-06	-0.012	-0.13	0.019	-0.024	0.0081	-0.011	-0.0051	0.021	-0.0082	-0.0019	0.021	-0.003	-0.14
price	-0.017	1	0.32	0.51	0.7	0.09	0.24	0.036	0.67	0.61	0.32	0.054	-0.053	0.31	0.022	0.59	0.082
bedrooms	0.0028	0.32	1	0.48	0.59	0.033	0.16	0.026	0.37	0.49	0.31	0.16	-0.16	-0.011	0.13	0.4	0.031
bathrooms	2.6e-06	0.51	0.48	1	0.7	0.085	0.48	-0.13	0.61	0.64	0.25	0.43	-0.16	0.031	0.18	0.51	0.081
sqft_living	-0.012	0.7	0.59	0.7	1	0.17	0.35	-0.059	0.76	0.88	0.44	0.32	-0.2	0.053	0.24	0.76	0.18
sqft_lot	-0.13	0.09	0.033	0.085	0.17	1	-0.0089	-0.0089	0.11	0.18	0.015	0.053	-0.13	-0.086	0.23	0.14	0.72
floors	0.019	0.24	0.16	0.48	0.35	-0.0089	1	-0.29	0.47	0.52	-0.24	0.58	-0.097	0.03	0.16	0.3	-0.013
condition	-0.024	0.036	0.026	-0.13	-0.059	-0.0089	-0.29	1	-0.14	-0.16	0.17	-0.36	0.003	-0.015	-0.11	-0.093	-0.0034
grade	0.0081	0.67	0.37	0.61	0.76	0.11	0.47	-0.14	1	0.76	0.17	0.45	-0.18	0.11	0.2	0.71	0.12
sqft_above	-0.011	0.61	0.49	0.64	0.88	0.18	0.52	-0.16	0.76	1	-0.052	0.42	-0.26	-0.00078	0.34	0.73	0.19
sqft_basement	-0.0051	0.32	0.31	0.25	0.44	0.015	-0.24	0.17	0.17	-0.052	1	-0.13	0.075	0.11	-0.14	0.2	0.017
yr_built	0.021	0.054	0.16	0.43	0.32	0.053	0.58	-0.36	0.45	0.42	-0.13	1	-0.35	-0.15	0.41	0.33	0.071
zipcode	-0.0082	-0.053	-0.16	-0.16	-0.2	-0.13	-0.097	0.003	-0.18	-0.26	0.075	-0.35	1	0.27	-0.56	-0.28	-0.15
lat	-0.0019	0.31	-0.011	0.031	0.053	-0.086	0.03	-0.015	0.11	-0.00078	0.11	-0.15	0.27	1	-0.14	0.049	-0.086
long	0.021	0.022	0.13	0.18	0.24	0.23	0.16	-0.11	0.2	0.34	-0.14	0.41	-0.56	-0.14	1	0.33	0.25
sqft_living15	-0.003	0.59	0.4	0.51	0.76	0.14	0.3	-0.093	0.71	0.73	0.2	0.33	-0.28	0.049	0.33	1	0.18
sqft_lot15	-0.14	0.082	0.031	0.081	0.18	0.72	-0.013	-0.0034	0.12	0.19	0.017	0.071	-0.15	-0.086	0.25	0.18	1

In [18]:

```
plt.scatter(house["sqft_living"],house["price"])
plt.xlabel("sqft_living")
plt.ylabel("price")
```

Out[18]:

```
Text(0, 0.5, 'price')
```



In [19]:

```
house["sqft_living"].value_counts()
```

Out[19]:

```
1300    138
1400    135
1440    133
1010    129
1660    129
       ...
3001      1
4970      1
2905      1
2793      1
1975      1
Name: sqft_living, Length: 1038, dtype: int64
```

In [20]:

```
house["grade"].value_counts()
```

Out[20]:

```
7     8979
8     6068
9     2615
6     2038
10    1134
11     399
5      242
12      90
4       29
13      13
3        3
1        1
Name: grade, dtype: int64
```

In [21]:

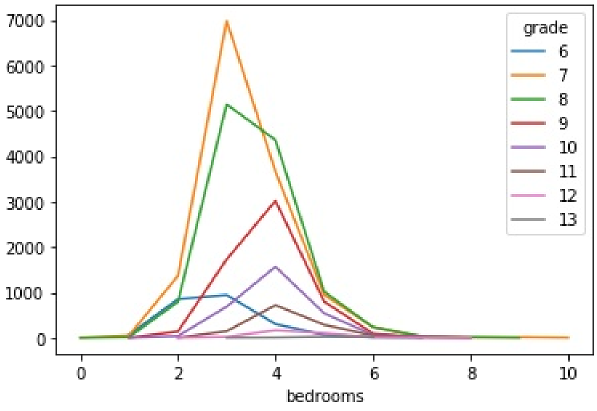
```
house=house.drop(house[(house.grade<6)].index)
```

In [22]:

```
house.pivot_table('bathrooms', index='bedrooms',columns='grade',aggfunc="sum").plot()
house.pivot_table('bathrooms', index='bedrooms',columns='grade',aggfunc="sum").plot()
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xce3f508>
```

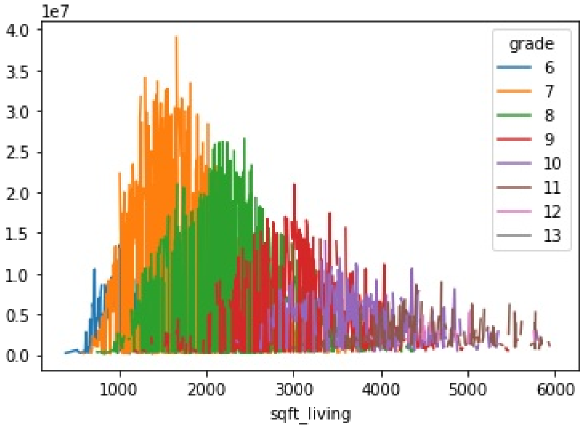


In [81]:

```
house.pivot_table('price', index='sqft_living',columns='grade',aggfunc="sum").plot()
```

Out[81]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xbb18108>
```



In [80]:

```
house=house.drop(house[(house.price>300000)&(house.sqft_living>6000)].index)
house.head()
```

Out[80]:

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	condition	grade	sqft_above	sqft_base
0	7129300520	20141013T000000	221900.0	3	1	1180	5650	1	3	7	1180	
1	6414100192	20141209T000000	538000.0	3	2	2570	7242	2	3	7	2170	
2	5631500400	20150225T000000	180000.0	2	1	770	10000	1	3	6	770	
3	2487200875	20141209T000000	604000.0	4	3	1960	5000	1	5	7	1050	
4	1954400510	20150218T000000	510000.0	3	2	1680	8080	1	3	8	1680	

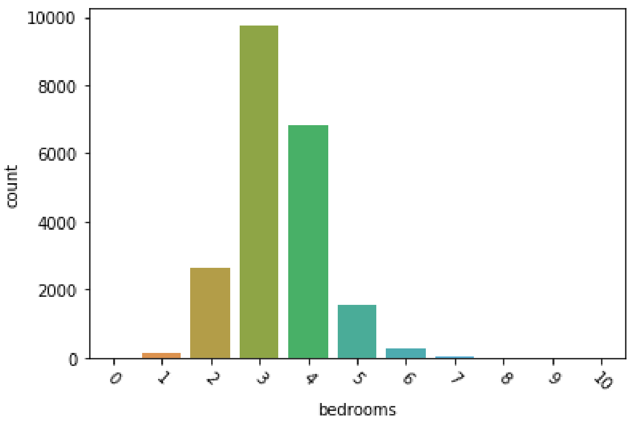
In [27]:

```
sns.countplot(x="bedrooms",data=house)
```

```
plt.xticks(rotation=-45)
```

Out[27]:

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10]),
 <a list of 11 Text xticklabel objects>)
```

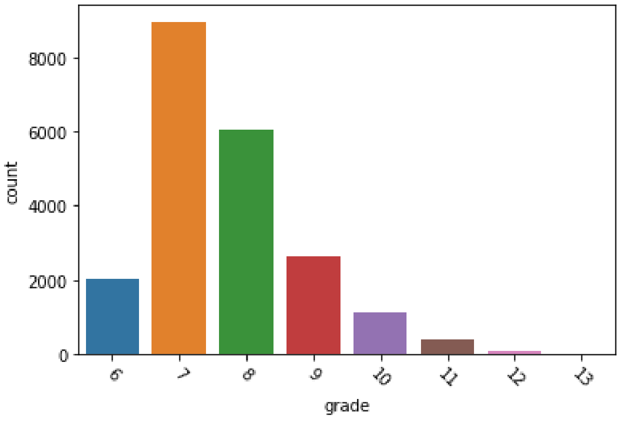


In [29]:

```
sns.countplot(x="grade",data=house)
plt.xticks(rotation=-45)
```

Out[29]:

```
(array([0, 1, 2, 3, 4, 5, 6, 7]), <a list of 8 Text xticklabel objects>)
```



Regression linéaire

In [30]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn import metrics
```

In [31]:

```
x=house["price"]
y=house["sqft_living"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=30)
```

In [48]:

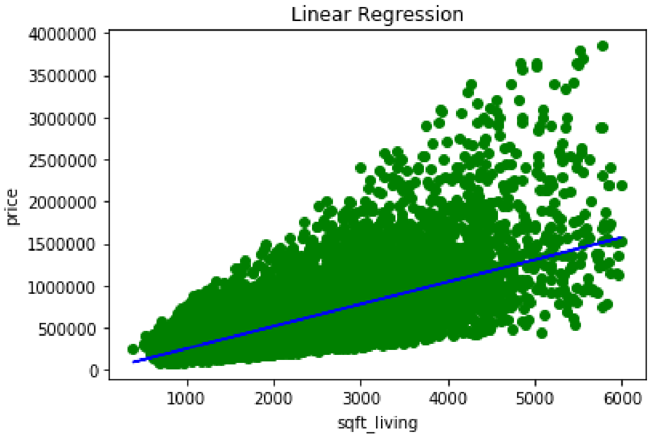
```
#extract x and y from our data
```

```
#extract x and y from our data
x=house["sqft_living"].values[:,np.newaxis]
y=house["price"].values

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.35,random_state=40) #splitting data
with test size of 35%

model=LinearRegression()    #build linear regression model
model.fit(x_train,y_train)  #fitting the training data
predicted=model.predict(x_test) #testing our model’s performance
print("MSE", mean_squared_error(y_test,predicted))
print("R squared", metrics.r2_score(y_test,predicted))
```

```
MSE 60164508415.7486
R squared 0.4680860370228618
```

In [33]:

```
plt.scatter(x,y,color="g")
plt.title("Linear Regression")
plt.ylabel("price")
plt.xlabel("sqft_living")
plt.plot(x,model.predict(x),color="b")
plt.show()
```



Regression Multilineaire

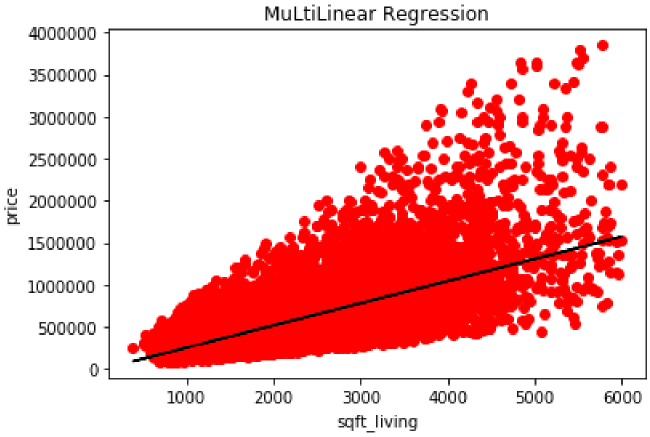
In [47]:

```
#Multilinear
#extract x and y from our data
x=house[["sqft_living","grade"]]
  #we have more than one input
y=house["price"].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.35,random_state=40) #splitting data
with test size of 35%

model=LinearRegression() #build linear regression model
model.fit(x_train,y_train) #fitting the training data
predicted=model.predict(x_test) #testing our model’s performance

print("MSE", mean_squared_error(y_test,predicted))
print("R squared", metrics.r2_score(y_test,predicted))
```

```
MSE 53467526289.49688
R squared 0.5272940052513393
```

In [45]:

```
plt.scatter(x, y, color="r")
plt.title("MuLtiLinear Regression")
plt.ylabel("price")
plt.xlabel("sqft_living")
plt.plot(x,model.predict(x),color="k")
plt.show()
plt.show()
```



Regression Polynomiale

In [46]:

```
#Importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

from sklearn.metrics import mean_squared_error
from sklearn import metrics
x= house[["sqft_living", "grade"]]
y= house["price"].values

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.35, random_state=40)  #splitting data
lg=LinearRegression()
poly=PolynomialFeatures(degree=3)

x_train_fit = poly.fit_transform(x_train) #transforming our input data
lg.fit(x_train_fit, y_train)
x_test_ = poly.fit_transform(x_test)
predicted = lg.predict(x_test_)

print("MSE: ", metrics.mean_squared_error(y_test, predicted))
print("R squared: ", metrics.r2_score(y_test,predicted))
```

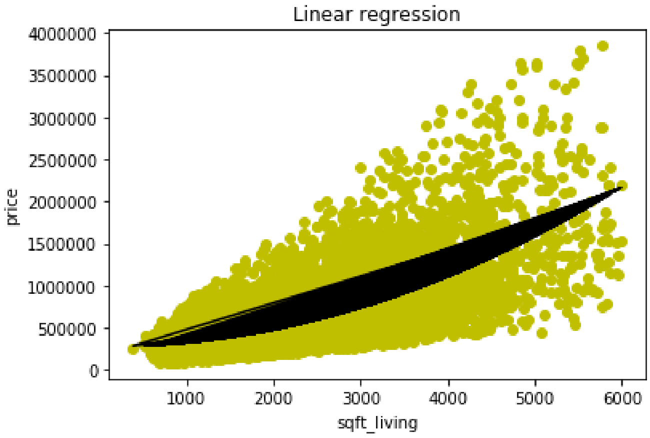
```
MSE:  49231222144.91536
R squared:  0.5647471380908822
```

In [79]:

```
x= house[["sqft_living"]]
y= house["price"].values.reshape(-1,1)
poly = PolynomialFeatures(degree = 2)
x_poly = poly.fit_transform(x)
poly.fit(x_poly, y)
lg=LinearRegression()
lg.fit(x_poly, y)

plt.scatter(x, y, color="y")
plt.title("Linear regression")
plt.ylabel("price")
plt.xlabel("sqft_living")
plt.plot(x, lg.predict(poly.fit_transform(x)), color="k")
```

Out[79]:

```
[<matplotlib.lines.Line2D at 0x12ca3208>]
```



In [59]:

```
print(x.shape)
```

```
(21268, 1)
```

In [60]:

```
print(y.shape)
```

```
(21268, 1)
```

In [ ]: